PENERAPAN MULTIPLE ANT COLONY SYSTEM (MACS) UNTUK PENYELESAIAN VEHICLE ROUTING PROBLEM WITH TIME WINDOWS (VRPTW)

A.B. Tjandrarini [1], Dedy Budianto [2]

Sekolah Tinggi Manajemen Informatika & Teknik Komputer Surabaya (STIKOM)
Email: asteriara@stikom.edu

Abstrak. Vehicle Routing Problem With Time Windows (VRPTW) is the develop from VRP problem that give armada capacity limit and each customer time window limit. Armada capacity is maximum limit for an armada carrying goods. Time window is a time range limit for request served by an armada. Metaheuristic is a searching method to solve complex combination problem with very large of solution search space. Differ with conventional method that concentrate to one solution only. In this research, we make a program with Multiple Ant Colony System (MACS) Algorithm to solve VRPTW problem.

***Keywords** : MACS, VRPTW, AS, ACS, CVRP*

Era globalisasi telah menuntut manusia untuk dapat mendistribusikan barang, sumber daya dan jasa ke segala penjuru dunia. Dalam hal ini transportasi menjadi kebutuhan manusia yang penting, karena transportasi dapat mendukung dan membuat semua aktifitas sosial dan ekonomi dapat berjalan. Transportasi sebagai sarana pengangkut sangat dibutuhkan untuk mendistribusikan barang ke tujuan yang diinginkan, misalnya ketika seseorang berangkat sekolah, pabrik mendistribusikan barang ke pengecer, kantor pos mengirimkan surat ke tujuan, dan semua kegiatan yang membutuhkan sarana pengangkut. Dari semua permasalahan transportasi itu maka yang perlu digaris bawahi adalah pentingnya permasalahan pemilihan jalan dan penjadwalan transportasi yang efektif untuk meminimalkan biaya.

Permasalahan untuk meminimalkan biaya dan total waktu biasanya disebut dengan *Vehicle Routing Problem* (VRP). VRP adalah suatu nama umum yang diberikan kepada suatu kelas permasalahan utuh dengan satu set rute untuk armada angkut (*vehicle*) yang berasal dari satu atau lebih depot yang harus disebarkan untuk melayani beberapa pelanggan (*customer*). Tujuan dari VRP adalah untuk melayani pelanggan sesuai dengan permintaannya dengan meminimalkan biaya angkut yang dimulai dan berakhir di depot.

VRP adalah permasalahan kombinasi yang komplek, yang dapat dilihat sebagai penggabungan dua permasalahan yang terkenal: *Traveling Salesman Problem* (TSP) dan *Bin Packing Problem* (BPP), yaitu diberikan suatu armada angkut dengan kapasitas seragam, suatu depot, dan beberapa pelanggan dengan permintaan (*demand*). Kemudian permasalahannya adalah mencari suatu rute yang dilalui dengan keseluruhan biaya rute yang minimal yang dapat melayani semua permintaan pelanggan. Semua perjalanan dimulai dan diakhiri di depot dan rute perjalanan armada harus dirancang sehingga masing-masing pelanggan hanya dilayani sekali dan hanya oleh satu armada. VRP adalah NP (*Non Polynomial*)-Hardness dan oleh karena itu sulit untuk memecahkannya, maka untuk menyelesaikan masalah VRP diperlukan suatu algoritma pendekatan yang dapat membantu memecahkan masalah ini dengan mudah.

Sampai saat ini metode untuk memecahkan permasalahan VRP dikategorikan atas tiga generasi. Generasi pertama adalah metode *Exact Approachs* yaitu algoritma yang mencari dan menghitung tiap-tiap kemungkinan solusi sampai salah satu solusi terbaik tercapai. Kelemahan dari metode ini adalah waktu pencarian solusi yang lama bila jumlah pelanggan yang dicari besar. Generasi kedua adalah metode *Heuristic* yaitu algoritma yang

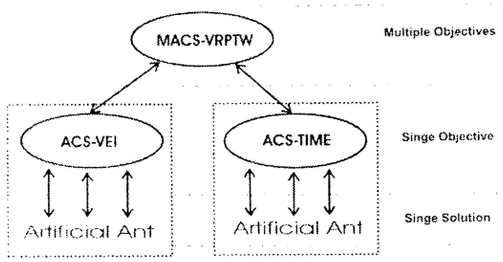
membangun suatu explorasi pencarian terbatas dan menghasilkan solusi yang baik dengan waktu yang cepat. Generasi ketiga adalah metode *Metaheuristic* merupakan pengembangan dari metode *heuristic*, dengan cara melakukan explorasi yang lebih dalam meningkatkan hasil solusi terbaik dari pendekatan *heuristic*, sehingga solusi yang dihasilkan jauh lebih baik dari pada hasil metode *heuristic*. *Multiple Ant Colony System* (MACS) adalah salah satu dari metode *Metaheuristic*.

ANT SYSTEM

Penggunaan *Ant System* (AS) untuk memecahkan permasalahan VRP pertama kali dibuat oleh Bullnheimer tahun 1997 yang diterapkan pada versi dasar dari problem ini, yaitu *Capacitated Vehicle Routing Problem* (CVRP). Kemudian penelitian tersebut dilanjutkan oleh Gambardella tahun 1999 yang menerapkan *Ant Colony System* (ACS) pada permasalahan VRPTW yang ditulis dalam paper berjudul "*Multiple Ant Colony System for Armada routing Problem with time Windows* " [3].

Dalam VRPTW terdapat dua sasaran tujuan yaitu: (i) meminimalkan jumlah perjalanan atau jumlah armada dan (ii) meminimalkan total waktu perjalanan, meminimalkan jumlah tour atau armada lebih diutamakan daripada meminimalkan total waktu perjalanan.

Dari Gambar 1 di bawah dapat diketahui bahwa arsitektur dari algoritma ini adalah akan menggunakan dua buah *artificial ant* yang berdasarkan pada algoritma ACS.



Gambar 2.1 Arsitektur dari MACS-VRPTW

Kedua koloni tersebut punya tujuan yang berbeda yaitu: koloni pertama ACS-VEI adalah mencoba mengurangi jumlah armada yang digunakan, kemudian koloni kedua ACS-TIME adalah mengoptimalkan solusi yang ditemukan oleh ACS-VEI.

Kedua koloni ini menggunakan *pheromone trail* yang terpisah, tetapi bekerjasama dengan sharing variable ψ^{gb} yang diatur oleh MACS-VRPTW

Solusi awal dari ψ^{gb} dibangun dengan menggunakan algoritma *nearest neighbor*. Kemudian ψ^{gb} akan ditingkatkan hasilnya dengan menggunakan dua buah koloni. Ketika ACS-VEI dijalankan, maka dia akan mencoba solusi yang mungkin dengan menggunakan armada dengan jumlah satu armada lebih sedikit dari jumlah armada yang digunakan oleh solusi ψ^{gb}.Tujuan dari ACS-TIME adalah mengoptimalkan lama waktu perjalanan dengan menggunakan sejumlah armada, yaitu sejumlah armada yang digunakan oleh solusi ψ^{gb}. ψ^{gb} akan diubah tiap-tiap waktu salah satu koloni menghasilkan solusi yang lebih baik. Pada saat suatu solusi lebih baik dengan jumlah armada lebih sedikit dari jumlah armada ψ^{gb} ditemukan, maka ACS-TIME dan ACS-VEI dihentikan dan kemudian proses diulang dan dua koloni baru diaktifkan, yang bekerja dengan jumlah armada baru.

METODE

Algoritma berikut ini merupakan inti dari algoritma MACS-VRPTW yang akan digunakan untuk menyelesaikan permasalahan VRPTW. Algoritma inilah yang menggabungkan antara koloni pertama dan koloni kedua. Pada penerapannya kedua koloni menggunakan dua buah *pheromone* yang terpisah dan tidak saling mempengaruhi.

Pada bagian inisialisasi, S_{best} adalah solusi terbaik (*global best solution*) dari algoritma MACS-VRPTW. Pertama kali S_{best} adalah sama dengan solusi awal yang dicari dengan menggunakan algoritma *Push Forward Insertion Heuristic* (PFIH). Solusi tersebut adalah solusi yang mungkin sehingga solusi ini akan dijadikan acuan seberapa baik solusi yang dihasilkan oleh kedua koloni. Variabel V pada

algoritma adalah jumlah armada dari solusi S_{best} saat ini.

Solusi S_{best} ditingkatkan kwalitasnya oleh kedua koloni. Pertama kali ACS-VEI diaktifkan, koloni ini mencoba untuk membangun solusi dengan harapan solusi yang dihasilkan adalah solusi dengan jumlah armada yang lebih sedikit satu armada dari V (yaitu jumlah armada S_{best} saat ini). Bila solusi yang diharapkan tercapai maka solusi S_{best} akan diganti dengan solusi yang dihasilkan oleh ACS-VEI.

Berikut diperlihatkan algoritma MACS-VRPTW.

Algoritma 1: *MACS-VRPTW*
Procedure MACS-VRPTW
1. /* *Inisialisasi* */
   $S \leftarrow$ Buat Solusi awal dengan PFIH
   $S_{best} \leftarrow S$
2. **Repeat**
   $V \leftarrow$ ActiveArmada(S_{best})
   Activate ACS-VEI(V - 1)
   Activate ACS-TIME(V)
   While ACS-VEI dan ACS-TIME active
   if ($S_{best} < S$) **then**
   $S_{best} \leftarrow S$
   if (ActiveArmada(S_{best}) < V) **then**
   kill ACS-TIME dan ACS-VEI
   end-while
   until kriteria berhenti
   Return S_{best}
End-procedure

ACS-TIME diaktifkan dengan tujuan mencoba untuk membangun solusi untuk meminimalkan total travel time/mengurangi travel time dari solusi S_{best} saat ini dengan menggunakan jumlah armada V. Apabila solusi yang dihasilkan ACS-TIME lebih baik dari solusi S_{best} maka solusi tersebut dijadikan solusi S_{best} sekarang dan tiap-tiap ada solusi yang lebih baik yang dihasilkan oleh kedua koloni tersebut maka ACS-VEI dan ACS-TIME dimatikan. Kemudian proses akan diulang lagi dan akan mengaktifkan koloni baru lagi. Proses tersebut akan dilakukan berulang-ulang sampai kriteria pemberhentian tercapai.

Berikut ini adalah implementasi dari algoritma MACS yang digunakan untuk menyelesaikan permasalahan VRPTW, yaitu merupakan penggabungan antara koloni VEHICLE dan koloni TIME.

```
Prosedur TMACS.MACS_VRPTW();
var
   Iteration      : Integer;
   ActiveVehicle : Integer;
begin
   Inisialisasi();
   InitSolution := TPFIH.Create( FGauge );
   InitSolution.PFIH;
   CopyRute( InitSolution.GBSolution,
GBSolution );
   Trail0 := 1 /
GBSolution.RuteTotal.TotalTravelTime;
   InitPheromone( Trail0, Pheromone_Vehicle
);
   InitPheromone( Trail0, Pheromone_Time );
   randomize;
   Iteration := 0;
   //construct solution
   repeat
      //update minimal vehicle yang mungkin
      ActiveVehicle :=
GBSolution.RuteTotal.TotalVehicle;
      //construct solution untuk meminimalkan
total vehicle
      if ActiveVehicle > 1 then
         ACS_VEHICLE( ActiveVehicle - 1 );
      //update minimal vehicle yang mungkin
      ActiveVehicle :=
GBSolution.RuteTotal.TotalVehicle;
      //construct solution untuk meminimalkan
travel time
      ACS_TIME( ActiveVehicle );
      Iteration := iteration + 1;
   until (Iteration = MaxIterMACS);
   //local search dengan 2-Interchange
   LamdaInterchange( 2, GBSolution.Rute,
GBSolution.RuteSum,
GBSolution.RuteTotal);
end;
```

ACS-VEHICLE merupakan koloni pertama dari algoritma MACS yang digunakan untuk meminimalkan jumlah armada yang digunakan. Implementasi dari koloni ini adalah sebagai berikut:

```
Prosedur TMACS.ACS_VEHICLE( const
vpVehicle : Integer );
var
  SolusiOK                    :
Boolean;
  i, k, Iteration, BestAnt, VisitedCustomer :
Integer;
  NodeIN          : Integer1D;
begin
  SetLength( NodeIN, VGcustomer + 1 );
  SolusiOK          := False;
  VisitedCustomer  := 0;
  Iteration          := 0;
  for i := 0 to vpVehicle-1 do
    VisitedCustomer := VisitedCustomer +
InitSolution.GBSolution.RuteSum[i].SumNode;
  repeat
    InisialisasiSemut();
    for k := 0 to NumAnt - 1 do begin
      NewActiveAnt( k, vpVehicle, NodeIN,
      Pheromone_Vehicle);
      for i := 1 to VGcustomer do
        if Semut[k].Visited[i] = 0 then
NodeIN[i] := NodeIN[i] + 1;
    end; //for k
    BestAnt := -1;
    for k := 0 to NumAnt - 1 do
     if ( Semut[k].Rute.RuteTotal.TotalNode
=
      VisitedCustomer ) or
      ( Semut[k].Rute.RuteTot al.TotalNode =
      VGcustomer ) then begin
      VisitedCustomer :=
        Semut[k].Rute.RuteTotal.TotalNode;
      if BestAnt = -1 then begin
        for i := 0 to VGcustomer do
NodeIN[i] := 0;
        BestAnt := k
      end
      else
      if ( BestRute2( Semut[ k ].Rute, Semut[
BestAnt
        ].Rute ) ) then
          BestAnt := k;
        end; //end if
    if (
Semut[BestAnt].Rute.RuteTotal.TotalNode =
VGcustomer ) and ( BestAnt <> -1 )
      then begin
        SolusiOK := True;
        CopyRute( Semut[BestAnt].Rute,
GBSolution );
      end;
      if ( BestAnt <> -1 ) then
        PheromoneGlobalUpdate(
Semut[BestAnt].Rute,
        Pheromone_Vehicle );
      PheromoneGlobalUpdate( GBSolution,
      Pheromone_Vehicle );
      Inc(Iteration);
  until ( Iteration = MaxIterVEHICLE ) or
(SolusiOK);
end;
```

Selanjutnya untuk implementasi Update Global Pheromone merupakan perubahan nilai *pheromone* berdasarkan pada solusi yang *global best*, sehingga perubahan hanya dilakukan setelah semua semut menyelesaikan membangun solusinya masing-masing. Implementasi dari perubahan nilai *pheromone* secara global adalah sebagai berikut:

```
Prosedur TMACS.PheromoneGlobalUpdate
( const vpRute : RuteStructure ; var
  Pheromone : Double2D );
var
  i, j, h : Integer;
begin
  for i := 1 to vpRute.RuteTotal.TotalNode +
    vpRute.RuteTotal.TotalVehicle do begin
    j    :=    vpRute.Rute[i-1].Node;    //node
sebelum
    h := vpRute.Rute[i].Node;  //node saat ini
    //Global pheromone update
    Pheromone[j][h] := ( 1 - Rho ) *
   Pheromone[j][h] + Rho /
   vpRute.RuteTotal.TotalTravelTime ;
  end;
end;
```

Pada prosedur diatas terdapat passing parameter *Pheromone*, ini diberikan karena dalam implementasi terdapat dua buah *pheromone* berbeda yang digunakan oleh tiap-tiap koloni, sehingga dengan passing parameter dapat digunakan satu prosedur untuk melakukan update untuk tiap *pheromone* yang berbeda.

HASIL DAN PEMBAHASAN

Ujicoba permasalahan VRPTW dalam penelitian ini digunakan jumlah node pelanggan yang harus dilayani sebanyak 25 pelanggan, dengan jumlah permintaan seluruh pelanggan sebanyak 460 dan untuk kapasitas daya angkut satu armada adalah 200. Untuk detail dari tiap-tiap pelanggan dapat dilihat pada Tabel 1 dan Tabel 2 berikut. Untuk pelanggan ke 0 adalah menandakan depot.

Tabel 1. Permasalahan VRPTW dengan 25 pelanggan (a)

Cust No.	X Position	Y Position	Demand
0	40	50	0
1	45	68	10
2	45	70	30
3	42	66	10
4	42	68	10
5	42	65	10
6	40	69	20
7	40	66	20
8	38	68	20
9	38	70	10
10	35	66	10
11	35	69	10
12	25	85	20
13	22	75	30
14	22	85	10
15	20	80	40
16	20	85	40
17	18	75	20
18	15	75	20
19	15	80	10
20	30	50	10
21	30	52	20
22	28	52	20
23	28	55	10
24	25	50	10
25	25	52	40

Tabel 2. Permasalahan VRPTW dengan 25 pelanggan (b)

Cust No.	Early Time	Latest Time	Service Time
0	0	1236	0
1	912	967	90
2	825	870	90
3	65	146	90
4	727	782	90
5	15	67	90
6	621	702	90
7	170	225	90
8	255	324	90
9	534	605	90
10	357	410	90
11	448	505	90
12	652	721	90
13	30	92	90
14	567	620	90
15	384	429	90
16	475	528	90
17	99	148	90
18	179	254	90
19	278	345	90
20	10	73	90
21	914	965	90
22	812	883	90
23	732	777	90
24	65	144	90
25	169	224	90

Setelah data permasalahan di atas dimasukkan dalam aplikasi kemudian dilakukan percobaan membentuk solusi sebanyak 5 kali percobaan, maka didapatkan Tabel hasil penyelesaian seperti pada Tabel 3 berikut.

Tabel 3. Hasil penelitian

Percobaan		MACS
1	V	**3**
	T	**2624**
	D	**295**
2	V	3
	T	2629
	D	259
3	V	3
	T	2633
	D	288
4	V	3
	T	2638
	D	332
5	V	3
	T	2629
	D	259
Keterangan:	V	: jumlah armada
	T	: total waktu perjalanan (*travel time*)
	D	: total jarak perjalanan (*travel distance*)

Dari Tabel 3 terlihat adanya solusi yang diberi tulisan tebal untuk menandai bahwa

solusi tersebut adalah solusi terbaik yang dapat ditemukan algorima MACS. Solusi ini menunjukkan bahwa jumlah armada dengan algoritma MACS adalah 3 armada. Hal ini sesuai dengan melihat dari permasalahannya bahwa jumlah permintaan pelanggan adalah 460 dan daya angkut satu armada adalah 200, maka jumlah armada yang dibutuhkan paling tidak adalah 460 dibagi 200 yang menghasilkan 3 armada. Selain itu menunjukkan total waktu perjalanan tercepat dengan total jarak perjalanan yang dibutuhkan.

Pada Tabel 4 berikut adalah hasil total dari pencarian solusi dengan menggunakan algoritma MACS dan Tabel 5 adalah urutan perjalanan dari tiap-tiap armada untuk melayani pelanggan.

Tabel 4. Total solusi MACS untuk 25 pelanggan

No Rute	Node	Demand	Travel Time	Travel Distance
1	5	100	529	79
2	10	170	1027	105
3	10	190	1068	112
Total	**25**	**460**	**2624**	**295**

Tabel 5. Detail solusi MACS untuk 25 pelanggan

No Rute	Urutan Perjalanan
1	0 – 5 - 3 – 7 – 8 – 15 – 0
2	0 – 13 – 17 – 18 – 19 – 11 – 9 – 6 – 23 – 22 - 21 - 0
3	0 – 20 – 24 - 25 – 10 – 16 – 14 – 12 – 4 – 2 – 1 – 0

SIMPULAN

Dari hasil penelitian ini menunjukkan bahwa algoritma *Multiple Ant Colony System* (MACS) dapat menghasilkan solusi permasalahan *Vehicle Routing Problem With Time Windows* (VRPTW) yang optimal dengan waktu yang cepat.

DAFTAR RUJUKAN

[1] Andrea Roli, Christian Blum dan Marco Dorigo, *ACO for Maximal Constraint Satisfaction Problems*, MIC'2001 – 4 th Metaheuristics International Conference, Porto, Portugal, Juli 16-20, 2001

[2] Dorigo, Marco dan Stützle, Thomas, 2004, *Ant Colony Optimization*, A Bradford Book, England.

[3] Gambardela, Luca Maria, Taillard, Eric and Agazzi, Giovanni, 1999, *MACS-VRPTW Multiple Ant Colony System for Vehicle Routing Problems With Time Windows*, Lugano, Switzerland.

[4] M. Dorigo. Optimization, Learning and Natural Algorithms (in Italian). PhD thesis, DEI, Politecnico di Milano,Italy, 1992. pp. 140.

[5] M. Dorigo and G. Di Caro. The Ant Colony Optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization, pages 11{32. McGraw-Hill, 1999.

[6] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. Art. Life, 5(2):137 (172), 1999.